重ねあわせを用いたビジュアルプログラミングに関する研究

奥村 穂高
Hotaka OKUMURA
(指導教官 山下 義行)

Abstract – Visual Proguramming is becoming more simple and practical than before with the progress of Graphical User Intarface (GUI). Visual Programming must be used "for beginners" and "user friendly."

This paper proposes an technique in Visual Programming that describes a new flowchart which combines two different flow at the same time. One is the data flow described by using "overlapping." The other is the control flow described by "directed edges." This paper also proposes a graphical programming environment in which the direct manipulation is possible. In addition, a prototype system is described in this paper.

1 はじめに

ビジュアルプログラミングとは、「いくつかの意味のある図的な表現をプログラミングの過程で使用すること」と定義される [1]。プログラミングは主に理論的、言語的な能力によって成立する活動だと思われがちだが、そこに、図的な表現を導入することによってパターン認知などが可能になり、直観的能力をより有効に活用することができる環境になる。このことは、プログラミングの効率化につながる。また、ビジュアルプログラミングは「複雑な事柄をシンボル化し、コンピュータにより直接操作できるものに置き換える技術」とも定義できる [2]。直接操作を利用することは、対象物に対して直に指示を反映させることができるため、扱う対象を認知のための負担が軽減され、プログラミングがより容易になる。

1980 年代に、デスクトップメタファーに基づいて作成された、 Apple 社のパーソナルコンピュータ・マッキントッシュが現れた。一方、ワークステーションでは、 UNIX 上で動作する 汎用ウインドウシステム X–Window System が MIT で開発、発表された。その後、それに基づいて Motif などの グラフィカルユーザインターフェイス（GUI）が登場してきた。現在では、アプリケーションのユーザーインターフェイスに GUI を利用するのは必然となり、マウスなどのポインティングデバイスを用いることによって、コマンドベースの場合より、高度な直接操作が可能となった。このことは、従来のコマンドベースのインターフェースから、よりヒューマンフレンドリーなインターフェースへと人々が求めるようになってきた結果といえよう。

GUI の発達により、ビジュアルプログラミングはより手軽で実用的な技術となった。今後、ビジュアルプログラミングの利点である「初心者向き」「ユーザフレンドリ」という点もより注目されるであろう。

本論文は、ビジュアルプログラミングを使うにあたっての作業効率をあげるために、コマンド等をアイコン化し、データフローに「重ねあわせ」を、制御フローに有向グラフを用いることにより、２つのフロー図を組み合わる記述法を提案する。また、それを利用した直接操作可能なグラフィカルなプログラミングの統合環境を提案し、プログラムの編集を１つの画面の中で行なえるようなプロトタイプシステムを作成、紹介する。

2　関連研究

ここ 20 年、プログラムを容易にする試みとしていろいろと行なわれてきた。その一つとしてプログラムの流れを表記するために フローチャート や PAD [3] などが考え出された。また、他方ではアイコンを使った Pict、HI–VISUAL、Show and Tell など [1] のシステムがある。本論文は双方の技術を活用、もしくは、改良したビジュアルプログラミングシステムとの位置づけを考えている。そこで本研究との関連性が深い、フロー図、Pict、HI–VISUAL について紹介し、その問題点について述べる。

2.1　データフロー図と制御フロー図

プログラムを表記するにあたって、有向グラフを活用する場合がある。その有向グラフは、いくつかのものに分類される。変数などの情報（データ）の流れを記述した「データフロー図」、作業の手順を中心に記述した「制御フロー図」などである。例えば、OMT[5] にある機能モデルではデータフロー図中心であり、制御フローはあくまで例外的扱いである。 JIS X 0121 におけるフローチャートでは、用途によって制御フロー、データフローを使い分けており、コンピュータのプログラミングの時は基本的に制御フローを用いている [4] 。

しかし、プログラムの作成時にはデータの流れと制御の流れとの関係が必要になる場合がある。本論文ではデータフロー、制御フローをを統合することにより、より直観的にプログラム構造を把握できる環境を提案する。

2.2　Pict システム

Pict システムは、ワシントン大学の Glinert と Tanimoto によって発表されたグラフィックス中心のプログラミングシステムである [6]。数字と短い補助文をのぞいて、すべてアイコンと有向線分で書き表され、Pascal や BASIC と同程度の言語水準の記述を行うことができる。しかし、言語構造や変数についての制約が強く、このため、彼らは論文で「初心者のプログラミング教育用としてはかなり有用であるが、それ以上の用途には大規模な拡張が必要となるであろう。」と述べている。

2.3　HI–VISUAL

HI–VISUAL は 広島大学の平川らによって提案されたアイコンによるプログラミンシステムである [7] [8] 。グラフ形式を用いず、通常、アイコンの短所として扱われる多様性を利用するところに特徴がある。アイコンは必ず物体をあらわしており、 2 種類のアイコンを重ね合わせる組合わせによって、幾つか成り立つ動作のなかから適当なものを一意に決定する。例えば、鉛筆のアイコンを drag し、紙のアイコンの上に重ねた場合、「書く」という意味に該当する処理が実行される。逆に、紙のアイコンを鉛筆の上に重ねると「包む」という意味が当てられる。

このシステムは一般のオフィス環境を想定したものであり、扱う対象の抽象度は高く、また、ループや分岐などの制御構造は用意されていない。このため、既存のプログラミング言語のコードレベルの記述や新たなシステムを構築するのは困難である。

3　本システムの概念

この節では本論文で提案する概念についてより詳しく述べる。

本論文では『重ねあわせ』という技術を用いて、既存のテキストによるプログラミング言語に近い記述が行える、再利用性の高いビジュアルプログラミング環境を提案する。また、通常、個別に扱われるデータフローと制御フローを共存させることによって、プログラムの構造をより直接的なものにする。

3.1　各種ノードの定義・変数の利用

画面構成は、コマンドやデバイスを示す カード型ノードと、処理の流れを示す エッジ（ 矢印 ）、そして、幾つかのノードやノード群をまとめてグ

ループ化するための プレートから成る。ノードは以下の 3 種類ある。

データの加工などの作業を示すノード
→ 「機能ノード」
入出力デバイスやファイルなどを示すノード
→ 「I/O ノード」
プログラム上に必要な構造を示すノード
→ 「構造ノード」

それぞれ黒、黄、赤の色のついた枠によって区別する（図 1）。各ノードの中央部分には、そのノードが保持する、コマンド名やファイル名を記述しておく。

変数については、円を基本としたマークで示す（ 後出・図 8参照 ）。利用方法は各種ノードと同じように扱う。

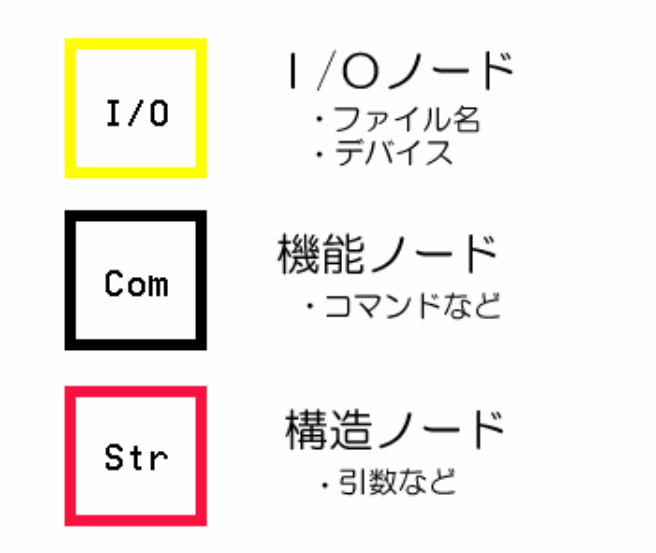

図 1: ノードの種類

3.2 重ねあわせ

本論文中での「重ねあわせ」とは、２つのノードを重ねることによって、その間にデータの流れを持たせるものと定義する (図 2)。重ねあわせを行った場合、データの流れは手前から奥の方へ流れていると見なす。データフローを重ねあわせで、制御フローをエッジ（矢印）で示すことによって２種類の流れを同時に表記することを可能にしている。

今回のシステムにおいては、以下のような場合にこの重ねあわせを使用している。

- 入出力デバイスの指定・変数の利用

  デバイスやファイルをノード化し、機能ノードに重ねあわせることによって、プログラムの入力に関する表記の簡略化が図れる。リダイレクトなどのデータが流れる場合がこれに含まれる (図 3)。また、変数についても同様に扱うことができる。

- フィルタリング

  １つのデータを複数の機能が連続的に加工する場合、重ね合わせを使うことによって途中経過を簡略化できる。 Shell Script で良く利用されるパイプはこのような状態である (図 4)。

- 引数

  引数は個別に構造ノードに書き、それを目的の機能ノードに重ねることにより実現する (図 4)。また、引数によって指定されるファイル名などは、接続する引数に更に重ねる形をとる。

図 3: リダイレクト

3.3 重ねあわせの利点・欠点

アイコンの重ねあわせは１入力１出力の場合にはプログラムの流れを把握するために大変有効である。本システムでは各ノードは重ねあわされているだけなので、１つのオブジェクトにまとめら

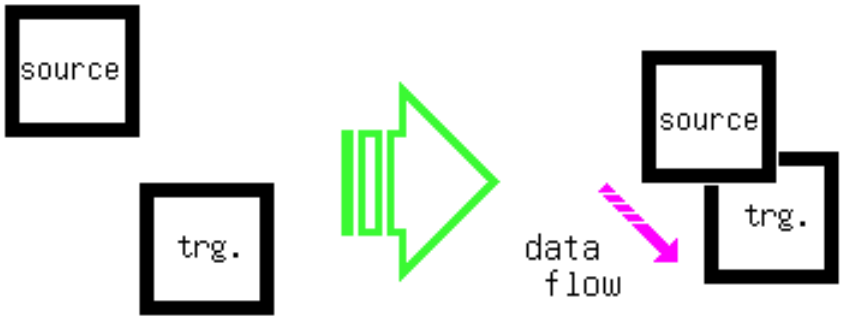

図 2: 重ねあわせ

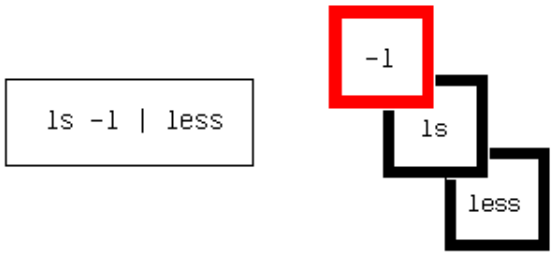

図 4: パイプ と 引数

れている場合と違い、中身を直接調べることもできる。このことは、直接操作を行ないやすくするために役立つ。

さらに、エッジの数を減らすことによって、グラフ構造を簡略化し、表示することができる。これにより、グラフ全体の把握や部分的なパターン認識の効率を向上できる。

しかし、レイアウトなどを考えずに重ね合わせると下のノードが隠れてしまい見えなくなる場合がある。また、ビジュアルプログラミングの共通の問題として、プログラミングを行なうために広いスペースが必要になる。

3.4 プレートによるグループ化

ある一連の処理をまとめて実施するときなどは、その該当する範囲を枠で囲み１つにまとめることによって、１つのノードと同等に扱うようにする。このように作られたものをプレートと呼ぶ。for、 while などのループは図 5 のように、機能ノードなどをプレートに重ねあわせることによって実現する。 shift 関数などもこれで対応する。

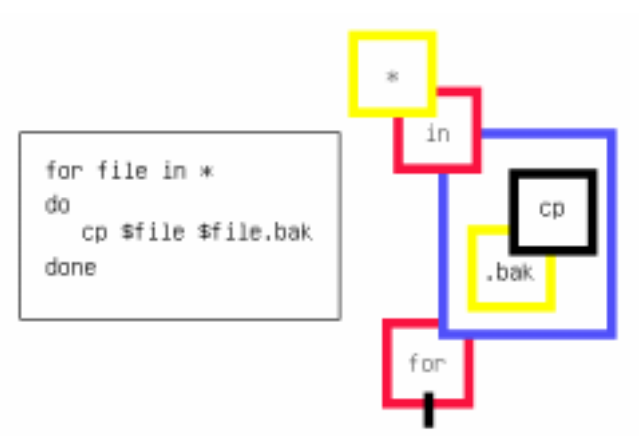

図 5: ループ

3.5 制御フローの記述

制御フローについては、エッジ（矢印）をつかった、フローチャートとほぼ同様な記述となる（図 6）。ただし、I/O ノード、変数、空のプレートを起点や終点にもつエッジは、それぞれの定義により、記述を禁止する。

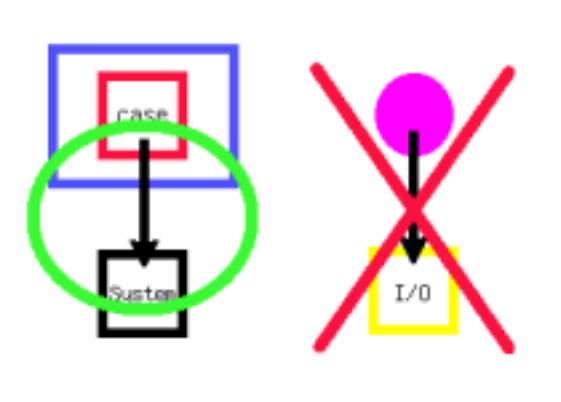

図 6: 制御フロー（左：正、右：誤）

3.6 システム構成・操作法

本システムは主にノードの種類やエッジなどを選択するウインドウと、選択したものを使って実際にプログラムを作成するウインドウにわかれる (図 7)。この２つのウインドウをつかい、プログラムに相当するグラフ構造を作成する。作成する手順としては次の通りである。

1. ノードの種類を選択する

   後で述べられる方法によってノードを作成する。

2. ノードを配置する

   制作側のウインドウの任意の位置に配置する。この際、重ね合わせによる関連付け（データフローの構成）ついても行われる。

3. フローを記述する・グループ化を行う

   選択側のウインドウにより、エッジ、プレートを選択し、分岐の作成、グループ化、制御フローの指定などを行う。

4. プログラムコードを出す

   １～３の手順を必要に応じて繰り返し、完成したプログラムに相当するグラフを解析しプログラムコードとしてファイル出力する。

3.7 ノード等の作成・移動

ノードは事前に選択肢側のウインドウで種類を選択する。その際、持たせるファイル名やコマンドを指定しておき、作成側のウインドウ内のノードの無い場所でクリックすることによって目的のノードを作成する。

プレートの場合もノードなどと同様に作成する。ただし、サイズが可変なため、サイズ決定には、カット＆ペーストを行なう場合と同様の操作で行なう。

クリックした先にノードや変数、プレートがあった場合、 drag and drop によってそれらを移動させることができる。

3.8 条件・分岐の記述

個々のノードや重ねあわせた一群の処理順序はエッジ（矢印）で表記する。エッジは、目的元のノードの下方から出て、目的先のノードの上方につながる。分岐の場合、真であるときは下から、偽であるときは右からのエッジに従う。また、 case 文などの条件節の場合、それぞれの構造ノードに記されている条件に該当する場合はそのノードの下につながるエッジに従う。どの条件も該当しない場合は、複数の構造ノードをまとめているプレートの右から出ているエッジに従う (図 8)。

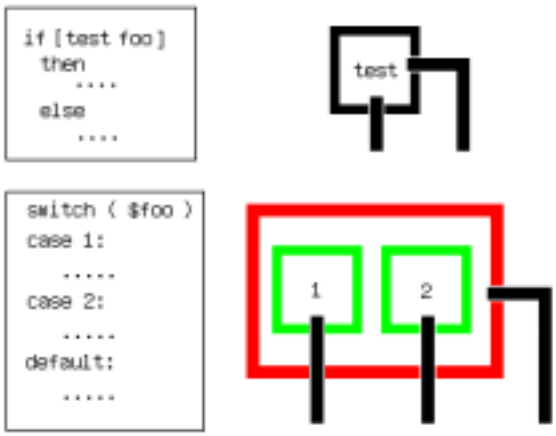

図 8: 条件節・多段分岐

3.9 開始条件・終了条件

プログラムの先頭は最上方・手前にあるノードであり、基本的にグラフの手前から奥に、上方

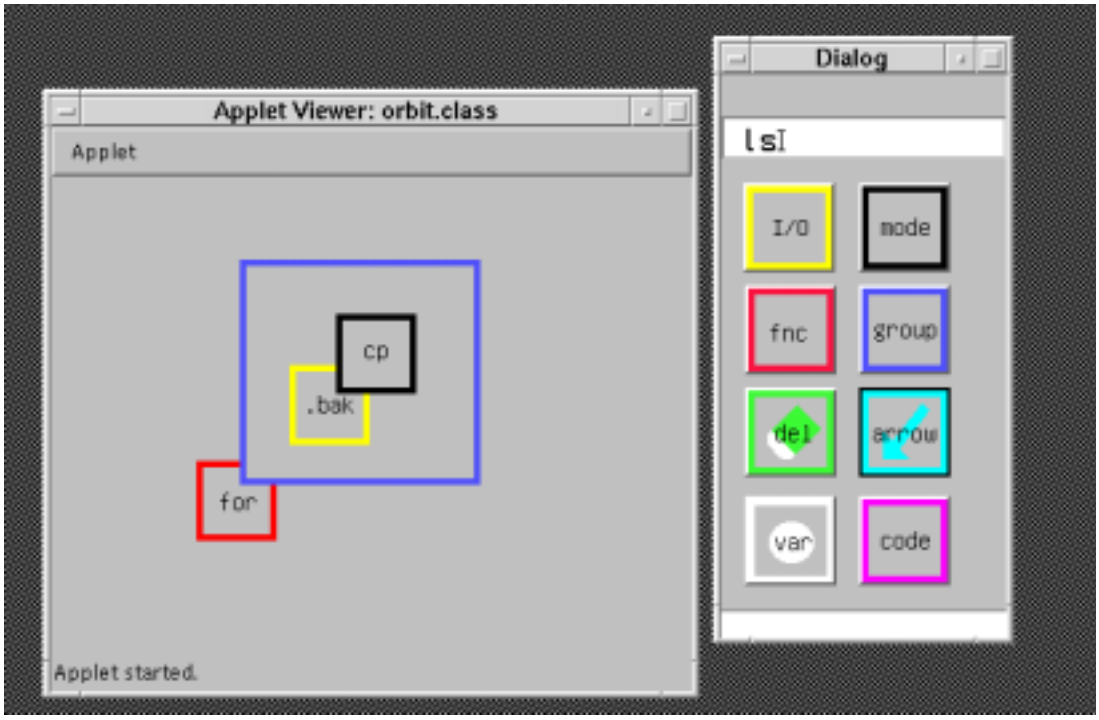

図 7: プロトタイプシステム

から下方に、左から右に向かってプログラムを追っていく形となる。また、次のノードに続かなくなった時点でそのプログラムは終了することとなる。

4　実装

4.1　実装の対象

前出の概念を Shell Script に適用する。Shell とは UNIX 上で使われるコマンドインタプリタであり、そこで使われるコマンドとその手順を記述した実行可能なプログラムを Shell Script という [9]。Shell コマンド は基本的に 1 入力 1 出力のため、重ねあわせを利用するのに適している。また、Shell Script の記述は横の流れがデータフロー、縦の流れが制御フローになっているのが特徴であり、本論文の概念に合致するので活用した。

本研究では Java を用いて Shell Script を視覚化することによって「重ねあわせ」などの概念を実装する。本論文を記述するにあたって、[10][11][12] を例を作る際の参考とした。

4.2　コマンドのパス

Shell Script では、コマンドのパスが重要な場合がある。本システムはノードにはコマンド本体を表記する。すべての機能ノードや I/O ノードは、特に指定が無い限り、そのプログラム構造を作成している環境においてのパス情報を保持している。パスの表記には吹き出しを使い、マウスカーソルが合わさっているノードのみ表記する（図 9）。

4.3　問題への対策

2 つのノードをそのまま重ねてしまうと、下のノードが隠れてしまうことがあり、ノードの内容が確認できなくなる場合がある。そのようなことを避けるために、ノードにはあらかじめ、「糊しろ」と呼ばれる部分を想定しておく。もし、糊しろをはみ出し、下にあるノードの中央部分に重なるような場合は、重ねようとするノード（drag し

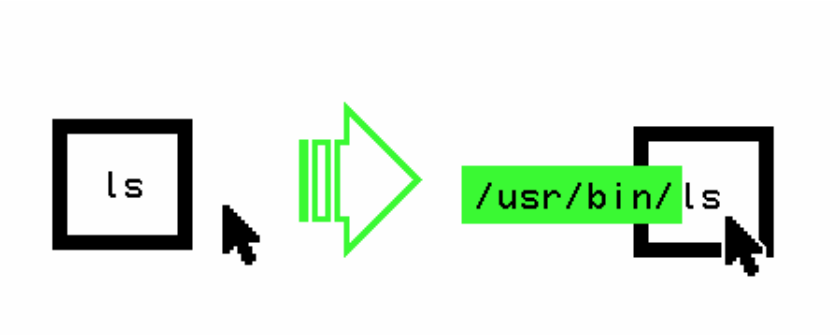

図 9: パスの記述

ていたノード）を現在位値から糊しろの部分まで移動することで、問題を解消する。

5 結論と今後の課題

本論文では、データフローに重ねあわせを、制御フローに有向グラフを用いることにより、この２つのフロー構造を同時に表記する手法を提案した。この手法を利用することにより、既存のビジュアルプログラミングをより直観的なものにすることができる。また、既存のテキストベースのプログラミング言語レベルの内容も記述できる。

提案した手法を用いた直接操作によるビジュアルプログラミング環境についても紹介した。このシステムは UNIX 上の Shell Script に対して利用し、Java によって実装した。「重ねあわせを用いた際に、ノードが隠れてしまう」という問題点を指摘して、その対処法として「糊しろ」を使うことを提案した。

今後、現状のシステムに存在する、以下のような問題点を解決していきたい。

- 変数の扱い
- プログラムの直接評価・トレースによるデバッグ
- 画面表示方法の工夫
- 他言語への応用

本論文の概念が持つ、他の既存システムに対しての有用性や実際の利用に際しての評価は不十分であり、これらを調査する必要がある。

6 謝辞

本研究にあたり、山下義行先生、田中二郎先生、中田育男先生、千葉 滋先生 の各先生方から貴重な御指導、助言等をいただきました。心より深く感謝致します。また、あらゆる方面から私を支えて下さいました 筑波大学プログラミング言語研究室、インタラクティブプログラミング研究室の両研究室の方々、相談にのって下さいました WISS'97 に参加されていた方々にもあわせて感謝致します。

参考文献

[1] Nan C.Shu, 訳： 西川博昭, ビジュアルプログラミング, 日経 BP 社 (1991)

[2] 田中二郎: ビジュアルプログラミング, ビジュアルインターフェース Bit ２月号別冊, pp.65–78, 共立出版 (1996)

[3] 河村一樹: PAD による構造化プログラミング, 啓学出版 (1988)

[4] 若山芳三朗, 吉川信之: 新しい JIS によるコンピュータのためのフローチャートの考え方・書き方, 啓学出版 (1987)

[5] J. ランボー, M. ブラハ, W. プレメニラ, F. エディ, W. ローレンセン, 監訳：羽生田栄一: オブジェクト指向方法論 OMT, トッパン (1992)

[6] Glinert, E., Tanimoto, S. : PICT: An Interactive Graphical Programming Environ-

ment, *IEEE Computer* , Vol.17, No.11, pp 7–25 (1984)

[7] Hirakawa, M., Tanaka, M. and Ichikawa, T. : An Iconic Programming System, HI–VISUAL, *IEEE Transaction on Software Engineering*, Vol.16, No.10, pp.1178–1184(1990)

[8] 市川忠男, 平川正人, かわりゆくプログラミング, 共立出版 (1994)

[9] 山口和紀 他, The UNIX Super Text [上], 技術評論社 (1992)

[10] UNIX System V プログラマ・リファレンス・マニュアル 第 2 版リリース 3.0, 共立出版 (1986)

[11] Lowell Jay Arther, 監訳：伊藤正安: UNIX シェルプログラミング, オーム社 (1993)

[12] エイチアイ: 実用 UNIX ツールハンドブック, ナツメ社 (1992)